# STAIR: High Reliable <u>ST</u>T-MRAM <u>A</u>ware Multi-Level <u>I</u>/O Cache Architectu<u>r</u>e by Adaptive ECC Allocation

Mostafa Hadizadeh, Elham Cheshmikhani, Hossein Asadi
*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*
mhadizdeh@ce.sharif.edu, elham.cheshmikhani@sharif.edu, asadi@sharif.edu

***Abstract*—*Hybrid Multi−Level Cache Architectures* (HCAs) are promising solutions for the growing need of high-performance and cost-efficient data storage systems. HCAs employ a high endurable memory as the first-level cache and a *Solid−State Drive* (SSD) as the second-level cache. *Spin−Transfer Torque Magnetic RAM* (STT-MRAM) is one of the most promising candidates for the first-level cache of HCAs because of its high endurance and DRAM-comparable performance along with non-volatility. However, STT-MRAM faces with three major reliability challenges named *Read Disturbance*, *Write Failure*, and *Retention Failure*. To provide a reliable HCA, the reliability challenges of STT-MRAM should be carefully addressed. To this end, this paper first makes a careful distinction between clean and dirty pages to classify and prioritize their different vulnerabilities. Then, we investigate the distribution of more vulnerable pages in the first-level cache of HCAs over 17 storage workloads. Our observations show that the protection overhead can be significantly reduced by adjusting the protection level of data pages based on their vulnerability. To this aim, we propose a *STT−MRAM Aware Multi−Level I/O Cache Architecture* (STAIR) to improve HCA reliability by dynamically generating extra strong *Error−Correction Codes* (ECCs) for the dirty data pages. STAIR adaptively allocates under-utilized parts of the first-level cache to store these extra ECCs. Our evaluations show that STAIR decreases the data loss probability by five orders of magnitude, on average, with negligible performance overhead (0.12% hit ratio reduction in the worst case) and 1.56% memory overhead for the cache controller.**

***Index Terms*—Data Storage Systems, Hybrid Multi-Level Cache Architecture, STT-MRAM, Error-Correction Code (ECC)**

## I. Introduction

In recent years, the over increasing rate of data generation necessitated new solutions for restructuring data storage systems [1]. Conventional *Solid-State Drive* (SSD)-caching schemes face with several challenges such as limited lifetime and performance degradation compared to the upper memory layers [2] [3]. *Hybrid Multi-Level Cache Architecture* (HCA) is one of the most promising approaches in terms of performance and cost efficiency. The main idea behind HCA, as shown in Fig. 1, is employing a more endurable (persistent) memory with lower latency as the first-level cache on top of the SSD, as the second-level cache. Therefore, the first-level cache bears most of the requests load, which leads to a considerable SSD lifetime improvement and higher performance compared to the conventional SSD caching schemes [4] [5].

Previous HCA schemes either exploit DRAM or *Phase Change Memory* (PCM) as the first-level cache [6] [7], while these schemes face with several challenges. The volatile property prevents DRAM-based HCAs from effective optimizations for write-intensive workloads without reliability degradation [4] [6]. Although PCM non-volatility provides an opportunity for more write optimizations, its limited endurance reduces cache lifetime besides degrading performance compared to DRAM-based HCAs [8] [9].

*Spin-Transfer Torque Magnetic RAM* (STT-MRAM) is one of the most promising candidates for the first-level cache in data storage systems. STT-MRAM provides the best of both worlds due to its non-volatility similar to PCM with significantly higher endurance and DRAM-comparable performance [10]–[12]. However, *none* of the previous studies take advantages of STT-MRAM in HCAs. Besides STT-MRAM advantages, it suffers from three major reliability challenges: 1) *Read Disturbance*, 2) *Write Failure*, and 3) *Retention Failure*. Thus, equipping HCAs with STT-MRAM requires overcoming these challenges to design a reliable HCA.

Previous studies on STT-MRAM reliability *either* proposed schemes for on-chip caches inside processors [13]–[18] *or* main memory [19]. These schemes, however, are *not* applicable to HCAs in storage systems. High volume of data requests makes scrubbing schemes [19] [20] unaffordable in these systems. On the other hand, employing software solutions such as checkpointing is not possible since the cache space is invisible to the upper layer, which assumes the data is reliably written to the back-end storage when delivering it to the cache. STT-MRAM-based HCAs require new reliability improvement solutions considering HCA characteristics, which is missing in the previous work.

In this paper, we propose a system-level low cost reliability enhancement scheme, named *<u>ST</u>T-MRAM <u>A</u>ware Multi-Level*

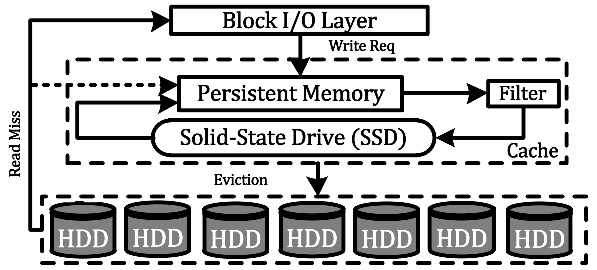

Fig. 1: Overview of hybrid multi-level I/O cache architecture (HCA)

*I/O Cache Architecture* (STAIR) for HCAs. To this end: 1) we first analyze the footprint of dirty data pages (which needs more protection against error) of well-known storage workloads on the first-level cache and then show that while the majority of workloads have small set of dirty data pages at the first-level cache, the *dirtiness* variation up to 97.35% is observed in some workloads. This observation implies the need for an adaptive solution with negligible storage overhead, and 2) motivated by the above investigation, we propose STAIR as the *first* STT-MRAM-aware HCA. STAIR provides higher protection capability for more vulnerable data pages of the first-level STT-MRAM cache to improve HCA reliability, by dynamically allocating parts of the cache to strong *Error-Correction Codes* (ECCs). Equipped with STAIR, as its names implies, this step-by-step ECC allocation approach increases HCA reliability without considerable storage overhead.

To evaluate STAIR, we develop a simulator equipped with DRAMSim2 [21] (configured with STT-MRAM cells [22]). Our evaluations over 17 well-known workloads from Microsoft research traces [23] [24], HammerDB [25], and Filebench [26], with dirtiness variation from 1.72% to 97.35%, show that the data loss probability is decreased by five orders of magnitude in STAIR with the cost of 0.12% reduction in hit ratio, in the worst case, and only 1.56% memory overhead for the cache controller.

The rest of this paper is organized as follows. Section II presents preliminaries of this work. In Section III, the challenges of employing STT-MRAM in HCA are discussed. The proposed STAIR scheme is explained in Section IV. In Section V, the evaluations and results are presented. Finally, Section VI concludes the paper.

## II. Preliminaries

### A. Hybrid Multi-Level Cache Architecture (HCA)

Equipping data storage systems with hybrid multi-level cache architecture, which utilizes a high endurance memory as the first-level cache and SSD as the second-level cache, results in a considerable SSD lifetime improvement [4] [7]. As shown in Fig. 1, the incoming page in a write request is admitted to the first-level cache in an ideal HCA and then acknowledgment is sent to the upper layer. A read miss causes reading the requested page from the main storage (HDD) and writing it to the first-level cache. Thus, the cache handles a majority of requests and significantly reduces the SSD writes, which improves the SSD lifetime [4]. In the case of a read hit at any level of HCA, the request is sent to the corresponding level and on a write hit, the previous version is invalidated (no matter where it was) and the request is admitted into the first-level cache. Some HCA schemes also employ a module to filter the insertion of an evicted data page from the first-level cache to the second-level cache to reduce SSD writes [2] [4] [6] [27].

The main drawback of previous HCA schemes is originated from the memory technologies employed for the first-level cache. Several DRAM-based schemes face with reliability degradation because of the volatility of DRAM cells. As an example, the scheme in [6] stores dirty data pages in DRAM, which leads to data loss in case of power outage. The risk of data loss in the case of power outage has forced DRAM-based schemes to increase the rate of SSD admission, which affects the SSD lifetime [4]. PCM-based schemes utilize the non-volatility of PCM [7] to alleviate the drawbacks of DRAM technology. However, PCM suffers from limited endurance and poor performance compared to DRAM. PCM-based schemes reduce HCA lifetime due to extreme load pressure on the first-level cache in addition to degrading the system performance due to its noticeable higher latency.

An efficient HCA requires a non-volatile memory for the first-level cache with a high endurance and DRAM-comparable performance. This scheme will be capable of permanently storing more number of requests without lifetime degradation, while providing high performance. Due to the several industrial and technical reports [10] [28] [29], STT-MRAM is the most promising technology for the first-level cache to meet the requirements of HCAs.

### B. STT-MRAM Basics

Similar to other non-volatile memory technologies, STT-MRAM consists of a storage element named *Magnetic Tunnel Junction* (MTJ) and an accesses element, which is an NMOS transistor. MTJ is formed from two ferromagnetic layers, *reference layer* and *free layer*, separated by a dielectric layer, named *tunnel barrier layer*. The content of the cell is based on the MTJ resistance. If the magnetic direction of the free layer compared to the reference layer is parallel, the resistance is low and the content is '0'. While the free layer spin direction is anti-parallel with the reference layer, the resistance is high, and the cell content is interpreted as '1' [30].

Although STT-MRAM provides promising features in terms of performance and endurance, it suffers from three error types: 1) *Read Disturbance*, 2) *Write Failure*, and 3) *Retention Failure*. The reading mechanism in the STT-MRAM is based on applying a small current to the free layer in order to distinguish the resistance of the cell. It is probable that this small current erroneously changes the cell content, which is called read disturbance [12]. Writing a new value to a STT-MRAM cell is based on applying a write pulse to the bit line or source line of the cell. Flowing the electronic charge from the reference layer to the free layer leads to parallelization of the free layer direction compared to the reference layer direction (logical value '0'). On the other hand, the current flowing from the reference layer to the free layer (electronic charge flowing from the free layer to the reference layer) antiparallelizes the direction of the free layer compared to the reference layer (logical value '1'). A write failure occurs if the cell content remains unchanged by current flowing in the cell during the write pulse [31]. Another type of error in the STT-MRAM cell is retention failure, which is a stochastic flip of the cell content without applying any read/write current to the MTJ [32]. Although STT-MRAM meets the requirements of the first-level cache (performance, endurance, and non-volatility),

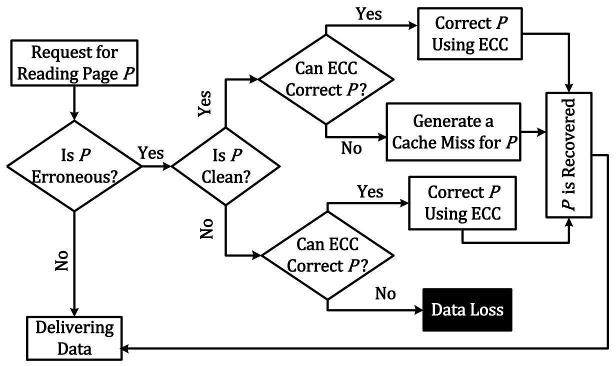

Fig. 2: Reading a data page from an ECC-protected first-level cache

the mentioned reliability challenges of the STT-MRAM should be carefully addressed to design an efficient HCA.

## III. STT-MRAM AS FIRST-LEVEL CACHE IN HCA

To design a reliable HCA, besides circuit- and architecture-level schemes, system-level considerations [31] [33] is a necessity for STT-MRAM-based HCAs. In this section, we first discuss the error correction codes as the conventional solutions for increasing the reliability of STT-MRAM-based HCAs. Then, we present our observations to illustrate the inefficiency of these solutions.

### *A. Conventional ECC-protected HCAs*

Utilizing redundancies such as parity, ECC, or data mirroring for data pages of the first-level cache increases the reliability of HCA. However, using a uniform redundancy for all data pages is an expensive approach due to different *criticality* of data pages in term of reliability. Employing a uniform data redundancy for all data pages occupies the usable cache space for storing data, which degrades the performance of the system and increases SSD writes. Therefore, a reliable HCA requires a cost-efficient redundancy approach with low storage overhead.

HCA should adaptively protect data pages according to their vulnerability. A main factor for determining the criticality of a data page is its dirtiness status. While there is a copy of clean data pages in the main storage, the dirty data pages in the first-level cache are the *only* valid copy of the data. Fig. 2 shows the operations performed on a read request based on probable scenarios for a data page. When a request for reading page $P$ is issued to the STT-MRAM-based first-level cache (protected by an internal ECC), while $P$ is an erroneous clean page, two scenarios can occur: 1) If the ECC of $P$ can correct the error, $P$ will be recovered, and 2) if the ECC is not capable to correct the page, the cache manager issues a cache miss to recover the page by the valid copy of $P$ in the main storage. In the case that erroneous $P$ is dirty, again we have two scenarios: 1) $P$ will be recovered if the ECC is capable to correct the error, and 2) the internal ECC is unable to correct the data and there is no valid copy of $P$, while the upper layer has already been acknowledged by the data storage. Hence, a data loss is occurred. Accordingly, to differentiate between data pages, the cache manager can bring the dirtiness/cleanness of each data page into account.

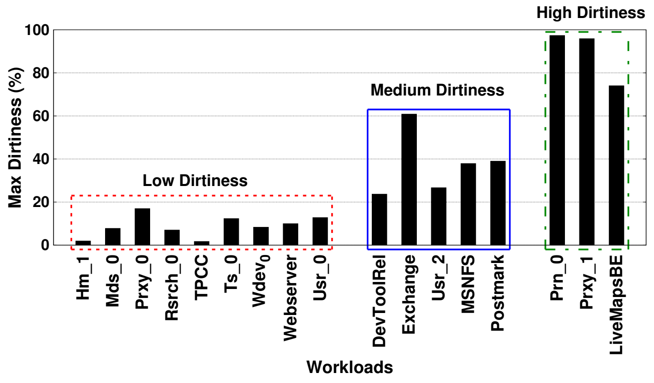

Fig. 3: Maximum dirtiness of the first-level cache

### *B. Observation and Motivation*

STT-MRAM is error-prone to the three aforementioned error types. Therefore, employing STT-MRAM as the first-level cache requires reliability-aware approaches for the dirty data pages to prevent data loss. Providing uniform redundancy for all data pages of the first-level cache is inefficient as it decreases the usable cache capacity. Therefore, HCA should exploit mechanisms to provide high protection for the dirty data pages of the first-level cache, while keeping the storage overhead as low as possible.

To explore the dirtiness in the first-level cache, we extract the maximum dirtiness of 17 workloads from Microsoft research traces [23] [24], HammerDB [25], and Filebench [26]. Defining *dirtiness* as the percentage of the number of dirty data pages to the total number of pages in the first-level cache, Fig. 3 shows the maximum dirtiness of the first-level cache while running each workload. The workloads are classified into three categories: 1) workloads with *low* dirtiness, 2) workloads with *medium* dirtiness, and 3) workloads with *high* dirtiness.

Low dirtiness category provides the possibility of stronger correction as the storage overhead of the redundancy is negligible compared to the total capacity of the first-level cache. For example, employing a mirroring technique for *Hm_1* leads to only 1.94% storage overhead while provides strong correction capability. This mechanism for medium and high dirtiness categories is inefficient as it can halve the cache space. HCA can use parity-based schemes to balance between the correction capability and storage overhead. Suppose that an approach is employed where a specific portion of the first-level cache has been allocated to the parity pages. A parity page is formed from three dirty data pages. Although this mechanism provides correction capability with lower storage overhead, it is still inefficient. For instance, the required storage for the parity pages of *Postmark* workload is estimated up to 9.75%, while about 15.25% of the allocated space will be idle.

HCA should take advantage of redundancies with low storage overhead to satisfy workloads from all three categories. To this aim, the allocated redundancy should be adaptievly adjusted based on the workload data criticality. In regard to

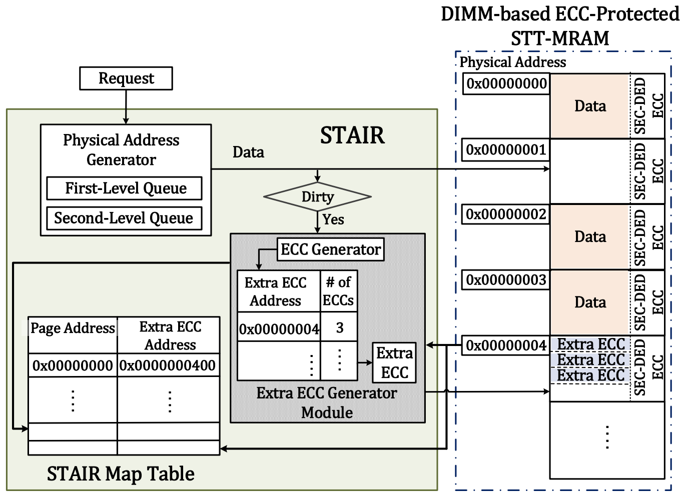

Fig. 4: Overview of STAIR structure

these observations, we propose a low-cost, highly reliable HCA scheme.

## IV. Proposed Method

HCAs equipped with STT-MRAM need efficient schemes to overcome STT-MRAM reliability challenges. To this end, we propose *STT-MRAM Aware Multi-Level I/O Cache Architecture* (STAIR), which its structure is shown in Fig. 4. Based on our investigations, the dirty pages are more critical in term of reliability, which need higher protection level. STAIR employs extra strong ECC for dirty data pages by allocating *idle* or *under-utilized* pages of the first-level cache to the required redundancy.

In the proposed scheme (depicted in Fig. 4), we assume that the first-level cache is a DIMM-based[1] STT-MRAM, in which all data pages are uniformly protected by an internal light-weight conventional (72,64) *Single Error Correction-Double Error Detection* (SEC-DED) code (64 bit data, 8 bit ECC). To adaptively allocate stronger ECCs to more critical data pages (dirty pages), STAIR uses two separate queues to manage each level of the cache. To write a data page to the first-level cache, STAIR generates the physical address based on the eviction policy. When the page is dirty, STAIR utilizes *Extra ECC Generator* module, which generates a (79,64) *Double Error Correction-Triple Error Detection* (DEC-TED) code for each dirty page.

To store the extra strong ECC, in the first step, STAIR exploits the byte-addressability provided by STT-MRAM and allocates under-utilized pages from the first-level cache to store extra ECC. STAIR first tries to fill out one of the already-allocated pages with the extra ECCs. To this end, the *Extra ECC Generator Module* keeps track of the ECC allocation status in each page and selects a page with free ECC slot, if any, based on *First Come, First Served* (FCFS) policy. In the case that no free ECC slot is found, STAIR moves to the next step and dynamically evicts the least utilized data page (based on the eviction policy) and stores the generated strong ECC in the newly allocated ECC page. STAIR employs a table named *STAIR Map Table* to track the corresponding extra ECC of each dirty data page. Each entry of STAIR Map Table consists of the physical address of both dirty data page and its corresponding extra ECC. In the case that the internal ECC is unable to correct the data, STAIR MAP Table is employed to access the strong ECC and recover the data.

When a dirty data page is evicted from the first-level cache, STAIR invalidates its corresponding extra ECC. If all extra ECCs in an ECC page become invalidated, the page is free and can be allocated to either data or extra ECCs. Thus, STAIR adaptively adjusts the number of extra ECC pages based on workload's read/write intensity and minimizes the storage overhead. In the case of corruption in a dirty data page, STAIR Map Table is referred to access the strong ECC for data recovery.

STAIR meets the requirements of a reliable HCA as it provides higher protection for dirty data pages. For 4KB page size, each page can store 34 extra ECCs. In the worst-case scenario, STAIR *only* occupies 2.86% of the first-level cache space for redundancy. Furthermore, no hardware modification is required for implementing this low-cost scheme and STAIR can be easily applicable to the *Commercial Off-The-Shelf* (COTS) devices.

## V. Evaluation

### A. Evaluation Methodology

In this section, the efficiency of STAIR is evaluated by developing a simulator for detailed analysis of the first-level cache of HCA. We equip our simulator with DRAMSim2 [21] and STT-MRAM cell configurations [22]. The simulated HCA consists of a 4GB first-level cache and a 120GB SSD (the second-level cache), while the page size is 4KB.

The HCA uses *Lazy Adaptive Replacement Cache* (LARC) [2] based on filtering, where the filtering is performed using a queue for tracking already-referenced pages of the first-level cache. This queue is capable of storing the last 90% accessed pages in the first-level cache. Each page is demoted from STT-MRAM to SSD if its metadata exists in the filtering queue. Moreover, both levels of the cache are managed based on the *Least Recently Used* (LRU) policy and each hit in the second-level cache leads to a promotion of the corresponding page. The same setup without applying STAIR to the first-level cache is used as the baseline for comparison.

All data pages in both the baseline and STAIR schemes are protected by SEC-DED(72,64), while STAIR uses DEC-TEC(79,64) as strong ECC for protecting data pages as well. Each 4KB data page is divided into 512 64-bit data words. The probability of data loss for each word protected by SEC-DED(72,64) is according to (1):

$$P_{data\_loss\_1word} = 1 - [P_{access}^{k} + \binom{k}{1} P_{access}^{k-1}(1 - P_{access})], \quad (1)$$

where $P_{access}$ is the probability of a successful access (read/write) of a single bit, which is $1-10^{-8}$ for read accesses and $1-10^{-7}$ for write accesses [12], [15], and *k* is the number

[1]Double In-line Memory Module

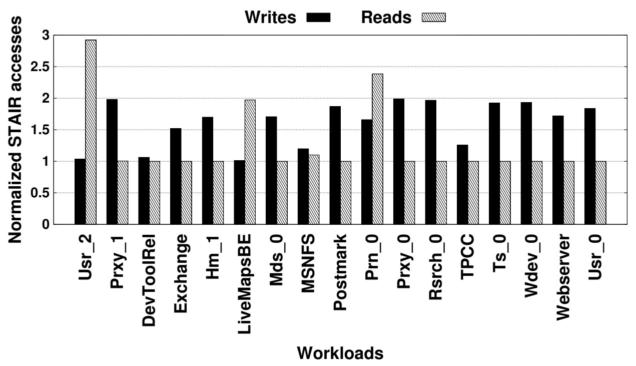

Fig. 5: Reads/Writes of STAIR normalized to the baseline

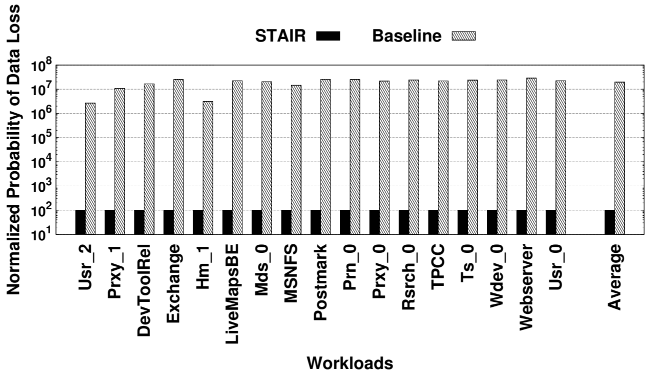

Fig. 6: Data loss probability of STAIR normalized to the baseline

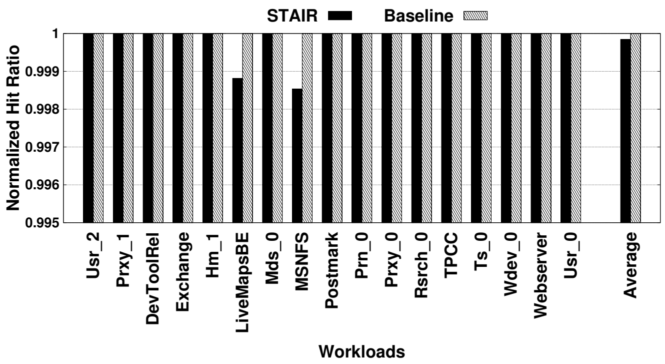

Fig. 7: Hit ratio of STAIR normalized to the baseline

of bits in a word, which is 64 in our experiments. Accordingly, the probability of data loss for a 4KB page is according to (2):

$$P_{data\_loss\_dirty\_page\_baseline} = (1 - [P_{access}^{k} + \binom{k}{1} P_{access}^{k-1}(1 - P_{access})])^{N}, \quad (2)$$

where *N* is the number of words, which is 512 for a 4KB page. Equipped with DEC-TED, STAIR decreases the probability of data loss using strong ECC to protect the dirty data pages. In this case, the probability of data loss for a data page in STAIR is according to (3):

$$P_{data\_loss\_dirty\_page\_STAIR} = P_{data\_loss\_clean\_page\_STAIR} = P_{data\_loss\_clean\_page\_baseline} = (1 - [P_{access}^{k} + \binom{k}{1} P_{access}^{k-1}(1 - P_{access}) + \binom{k}{2} P^{k-2}(1-P)^{2}])^{N} \quad (3)$$

According to aforementioned equations, the read/write stress of the first-level cache can significantly affect the probability of data loss. For clean data pages, STAIR and the baseline provides the same reliability as they both employ the same approach (recovering a correct version of data from the main storage in the case of uncorrectable error in the first-level cache). Fig. 5 shows the normalized number of reads and writes committed by STAIR to the first-level cache, compared to the baseline. Our simulator evaluates the reliability of the baseline and STAIR using the above equations, based on the read/write stress. Evaluations are performed on 17 storage workloads from Microsoft research traces [23] [24], HammerDB [25], and Filebench [26].

## *B. Reliability Evaluation*

STAIR reduces the probability of data loss by providing higher protection for more vulnerable data pages. By allocating some underutilized first-level cache pages to extra strong ECCs for dirty data pages, STAIR is capable of correcting higher number of errors, which improves the reliability of HCA. The simulator calculates the reliability for the baseline according to (4):

$$R_{Baseline} = (1 - P_{data\_loss\_dirty\_page\_baseline})^{M_d} \times (1 - P_{data\_loss\_clean\_page\_baseline})^{M_c}, \quad (4)$$

where $M_d$ and $M_c$ are the total number of accesses to the dirty and clean pages, respectively, and $R_{Baseline}$ is the reliability of the baseline. On the other hand, the reliability of STAIR is evaluated according to (5):

$$R_{STAIR} = (1 - P_{data\_loss\_dirty\_page\_STAIR})^{M_c + M_d}, \quad (5)$$

where $P_{data\_loss\_dirty\_page\_STAIR}$ is equal to the $P_{data\_loss\_clean\_page\_STAIR}$, since STAIR can correct two errors in a dirty/clean page.

Fig. 6 shows the normalized data loss probability of STAIR compared to the baseline. STAIR reduces the probability of data loss by five orders of magnitude, on average, compared to the baseline. This value is more than four orders of magnitude in the worst case. The results show that even in write-intensive workloads with a large amount of dirty pages, a high level of reliability is achieved by STAIR as it can correct two errors in a page, if occur.

## *C. Overheads*

STAIR exploits the least-utilized pages in the first-level cache to store the extra ECCs and each allocated page is capable of storing ECCs for multiple dirty data pages. Hence, the number of ECC pages is not considerable and the effect of STAIR on the cache hit ratio and storage performance is minimal. Fig. 7 shows the effect of STAIR on cache hit ratio. STAIR reduces the hit ratio by 0.02% and 0.12%, on average and in the worst case, respectively. As STAIR dynamically occupies the under-utilized pages of the first-level cache for the extra ECCs on demand, its hit ratio degradation is negligible.

In the case that STAIR evicts a data page for allocating the memory space to extra ECCs, the victim page is the least-recently used data page, which has the lowest chance to be accessed again. The higher hit ratio reduction in *MSNFS* and *LiveMapsBE* workloads depicts that the probability of re-accessing to the victim pages is higher than the other workloads, which results into performance degradation.

Exploring cache controller memory overhead, STAIR increases the cache controller memory size by only 1.56% for an HCA with 4GB first-level cache and 120GB second-level cache. The memory overhead evaluations are performed based on investigations of [34]. This overhead is imposed mainly for storing STAIR Map Table.

## VI. Conclusion

STT-MRAM technology is a promising alternative for PCM and DRAM technology in the first-level cache of HCAs in storage systems. However, the unreliability of STT-MRAM is a main limitation factor due to its high error rate. This paper proposed STAIR, a system-level scheme to efficiently protect STT-MRAM first-level cache in HCAs. By classifying data pages based on their vulnerability, STAIR adaptively provides higher protection level for more vulnerable pages. Compared to the uniform data page protection in conventional HCAs, STAIR reduces the data loss probability by an average of five orders of magnitude. This significant reliability enhancement is provided with a negligible overhead.

## References

[1] J. Niu, J. Xu, and L. Xie, “Hybrid Storage Systems: A Survey of Architectures and Algorithms,” *IEEE Access*, vol. 6, Feb. 2018.

[2] S. Huang, Q. Wei, D. Feng, J. Chen, and C. Chen, “Improving Flash-based Disk Cache with Lazy Adaptive Replacement,” ACM Trans. Storage (TOS), vol. 12, no. 2, Feb. 2016.

[3] R. Salkhordeh, S. Ebrahimi, and H. Asadi, “ReCA: An Efficient Reconfigurable Cache Architecture for Storage Systems with Online Workload Characterization,” IEEE Trans. Parallel Distrib. Syst. (TPDS), vol. 29, no. 7, Jul. 2018.

[4] R. Salkhordeh, M. Hadizadeh, and H. Asadi, “An Efficient Hybrid I/O Caching Architecture Using Heterogeneous SSDs,” IEEE Trans. Parallel Distrib. Syst. (TPDS), vol. 30, no. 6, Jun. 2019.

[5] X. Hu, X. Wang, L. Zhou, Y. Luo, Z. Wang, C. Ding, and C. Ye, “Fast Miss Ratio Curve Modeling for Storage Cache,” ACM Trans. Storage (TOS), vol. 14, no. 2, April 2019.

[6] D. Jiang, Y. Che, J. Xiong, and X. Ma, “uCache: A Utility-aware Multilevel SSD Cache Management Policy,” in Proc. of Int. Conf. High Perform. Comput. Commu. & IEEE Int. Conf. Embed. Ubiq. Comp. (HPCC_EUC), Nov. 2013, pp. 391-398.

[7] L. Shi, J. Li, C. J. Xue, and X. Zhou, “Hybrid Non-Volatile Disk Cache for Energy-efficient and High-performance Systems,” ACM Trans. Des. Auto. of Elect. Syst. (TODAES), vol. 18, no. 1, Jan. 2013.

[8] Reza Salkhordeh and H. Asadi, “An Operating System Level Data Migration Scheme in Hybrid DRAM-NVM Memory Architecture,” in Proc. of Conf. Des., Automat. Test Eur. (DATE), Mar. 2016, pp. 936-941.

[9] M. Tarihi, H. Asadi, A. Haghdoost, M. Arjomand, and H. Sarbazi-Azad, “A Hybrid Non-Volatile Cache Design for Solid-State Drives Using Comprehensive I/O Characterization,” IEEE Trans. Comput. (TC), vol. 65, no. 6, 2016.

[10] J. J. Kan et al., “A Study on Practically Unlimited Endurance of STT-MRAM,” IEEE Trans. Electronic Device (TED), vol. 64, no. 9, Sept. 2017.

[11] E. Kltrsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, “Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative,” in Proc. of IEEE Int. Symp. Perf. Analys. Syst. Softw. (ISPASS), April 2013, pp. 256-267.

[12] E. Cheshmikhani, H. Farbeh, and H. Asadi, “Enhancing Reliability of STT-MRAM Caches by Eliminating Read Disturbance Accumulation,” in Proc. of Conf. Des., Automat. Test Eur. (DATE), Mar. 2019, pp. 854-859.

[13] E. Cheshmikhani, H. Farbeh, and H. Asadi, “ROBIN: Incremental Oblique Interleaved ECC for Reliability Improvement in STT-MRAM Caches,” in Proc. of Asia South-Pacific Des. Automat. Conf. (ASP-DAC), Jan. 2019, pp. 173-178.

[14] M. Gupta et al., “Reliability-Aware Data Placement for Heterogeneous Memory Architecture,” in Proc. of IEEE Int. Symp. High Perform. Comput. Arch. (HPCA), Feb. 2018, pp. 583-595.

[15] E. Cheshmikhani, H. Farbeh, S.G. Miremadi, and H. Asadi, “TA-LRW: A Replacement Policy for Error Rate Reduction in STT-MRAM Caches,” IEEE Trans. Comput. (TC), vol. 68, no. 3, Mar. 2019.

[16] Y. Luo et al., “Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory,” in Proc. of IEEE/IFIP Int. Conf. Depen. Syst. and Net. (DSN), Jun. 2014, pp. 467-478.

[17] E. Cheshmikhani, A. M. Hosseini Monazah, H. Farbeh, and S. G. Miremadi, “Investigating the Effects of Process Variations and System Workloads on Reliability of STT-RAM Caches,” in Proc. of Euro. Depen. Comp. Conf. (EDCC), Sep. 2016, pp. 120-129.

[18] M. Kishani, M. Tahoori, and H. Asadi, ”Dependability Analysis of Data Storage Systems in Presence of Soft Errors,” IEEE Trans. Rel. (TR), vol. 68, no. 1, Mar. 2019.

[19] X. Guo, M. N. Bojnordi, Q. Guo, and E. Ipek, “Sanitizer: Mitigating the Impact of Expensive ECC Checks on STT-MRAM Based Main Memories,” IEEE Trans. Comput. (TC), vol. 67, no. 6, Jun. 2018.

[20] P. J. Nair, B. Asgari, and M. K. Qureshi, “SuDoku: Tolerating High-Rate of Transient Failures for Enabling Scalable STTRAM,” in Proc. of IEEE/IFIP Int. Conf. Depen. Syst. and Net. (DSN), Jun. 2019, pp. 388-400.

[21] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, “DRAMSim2: A Cycle Accurate Memory System Simulator,” IEEE Comp. Arch. Let. (CAL), vol. 10, no. 1, Mar. 2011.

[22] K. Asifuzzaman, R. S. Verdejo, and P. Radojkovi, “Enabling a Reliable STT-MRAM Main Memory Simulation,” in Proc. of ACM Int. Symp. Mem. Syst. (MEMSYS), Oct. 2017, pp. 283-292.

[23] D. Narayanan, A. Donnelly, and A. Rowstron, “Write Off-Loading: Practical Power Management for Enterprise Storage,” ACM Trans. Storage (TOS), vol. 4, no. 3, Nov. 2008.

[24] Storage Networking Industry Association, Microsoft enterprise traces. [Online]. Available: http://iotta.snia.org/traces/130, Accessed: 201908-10.

[25] S. Shaw, HammerDB: The Open Source Oracle Load Test Tool. [Online]. Available: http://www.hammerdb.com/, Accessed: 201908-10.

[26] V. Tarasov, E. Zadok, and S. Shepler, “Filebench: A Flexible Framework for File System Benchmarking,” USENIX; Login, vol. 41, 2016.

[27] Y. Ni, J. Jiang, D. Jiang, X. Ma, J. Xiong, and Y. Wang, “S-RAC: SSD Friendly Caching for Data Center Workloads,” in Proc. of ACM Int. Syst. Storage Conf. (SYSTOR), Jun. 2016, pp. 1-8.

[28] TSMC Corporation, Next generation MRAM. [Online] Available: https://www.tsmc.com/english/dedicatedFoundry/technology/eflash.htm, Accessed: 2019-08-25.

[29] Toshiba Corporation, Solid-state magnetic memory. [Online] Available: https://www.toshiba.co.jp/rdc/rd/topics_e_16.htm, Accessed: 2019-08-25.

[30] H. Naeimi, C. Augustine, A. Raychowdhury, S. L. Lu, and J. Tschanz, “STT-MRAM Scaling and Retention Failure,” Intel Tech. J. (ITJ), vol. 17, no. 1, 2013.

[31] E. Aliagha, A. M. H. Monazzah, and H. Farbeh, “REACT:Read/Write Error Rate Aware Coding Technique for Emerging STT-MRAM Caches,” IEEE Trans. Mag. (TMAG), vol. 55, no. 5, May 2019.

[32] E. Cheshmikhani, H. Farbeh, and H. Asadi, “A System-Level Framework for Analytical and Empirical Reliability Exploration of STT-MRAM Caches,” IEEE Trans. Rel. (TR), In Press, 2019.

[33] S. M. Nair, R. Bishnoi, and M. B. Tahoori, “A Comprehensive Framework for Parametric Failure Modeling and Yield Analysis of STT-MRAM,” IEEE Trans. VLSI (TVLSI) Syst., vol. 27, no. 7, Jul. 2019.

[34] Z. Chen, N. Xiao, Y. Lu, and F. Liu, “Me-CLOCK:A Memory-Efficient Framework to Implement Replacement Policies for Large Caches,” IEEE Trans. Comput. (TC), vol. 65, no. 8, Aug. 2016.